WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Wednesday, January 05, 2005

Hide?	Set Name	Query	Hit Count
		DB=USPT,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR	
☐	L31	external$4 near3 memory near3 setup near3 (data or information)	9
☐	L30	L28 and (setup near2 (data or information))	0
☐	L29	L28 same (setup near2 (data or information))	0
☐	L28	(embed$4 near2 processor) with (external$4 near2 memory)	28
☐	L27	L26 same external$4	3
☐	L26	boot$5 with (embed$4 near2 process$4)	22
☐	L25	l1 and (embed$4 near2 process$4)	2
☐	L24	l1.clm. and l16	18
☐	L23	l1.ti.	3
☐	L22	l1.ti. and L16	0
☐	L21	l8.ti. and L16	5
☐	L20	l8.clm. and L16	36
☐	L19	l8 and L16	73
☐	L18	l6 and L16	20
☐	L17	l1 and L16	35
☐	L16	l14 or L15	2672
☐	L15	709/220,222.ccls.	1098
☐	L14	713/1,2.ccls.	1718
☐	L13	initializ$9 with (without near3 (external$4 or outside) near3 (access$4 or memory or storage))	5
☐	L12	initializ$9 same (without near3 (external$4 or outside) near3 (access$4 or memory or storage))	19
☐	L11	(internal$4 near3 initializ$9) same (without near3 (external$4 or outside) near3 (access$4 or memory or storage))	0
☐	L10	L1 same ((local$4 or inside or internal$4 or on-board) near3 (fifo or register or memory or cache or storage or buffer or queue))	0
☐	L9	L1 and ((local$4 or inside or internal$4 or on-board) near3 (fifo or register or memory or cache or storage or buffer or queue))	25
☐	L8	(non-volatile near3 boot$5 near3 (memory or storage or fifo or buffer or queue or cache))	217
☐	L7	(execut$4 near3 (portion or partial$4 or section or segment) near3 bootstrap$5)	5

☐	L6	L1 and ((local$4 or inside or internal$4 or on-board) near3 (memory or cache or storage or buffer or queue))	24
☐	L5	L1 same ((local$4 or inside or internal$4 or on-board) near3 (memory or cache or storage or buffer or queue))	0
☐	L4	L1 same ((local$4 or inside or internal$4) near3 (memory or cache or storage or buffer or queue))	0
☐	L3	L1 same (internal$4 near3 (memory or cache or storage or buffer or queue))	0
☐	L2	L1 same ((using or utilizing) near3 internal$4 near3 (memory or cache or storage or buffer or queue))	0
☐	L1	(execut$4 near3 (portion or partial$4 or section or segment) near3 boot$5)	86

END OF SEARCH HISTORY

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L27: Entry 2 of 3 File: USPT Oct 9, 2001

DOCUMENT-IDENTIFIER: US 6301656 B1
TITLE: Method and apparatus for initial programming of flash based firmware

Brief Summary Text (5):
In a typical operation, on power up, a control board, including a microprocessor, executes its on-board programming which includes a setup protocol for establishing communication with components external to the control board. Once communication is established, the microprocessor can receive a new programming load. This programming is written onto the flash memory. When a board with an embedded processor is originally built, the flash memory must be programmed with a boot code so that the board can establish communication with the outside world. This is typically done by taking unprogrammed flash memories and placing them on a programmer. They are then programmed and replaced on the board. The board is then booted and the programming can be loaded from the flash memory to the microprocessor. If a new boot code is required, the flash devices are removed, reprogrammed and then reinstalled. To allow for removal, flash memories are typically installed in sockets instead of being soldered onto the board. The handling of flash memory can damage its metal leads or produce dangerous static electricity which can damage the memories. Also, using a programmer to program flash memory is a slow process.

Previous Doc Next Doc Go to Doc#

L27: Entry 1 of 3 File: USPT Oct 12, 2004

DOCUMENT-IDENTIFIER: US 6803785 B1
TITLE: I/O circuitry shared between processor and programmable logic portions of an integrated circuit

Detailed Description Text (45):
Reset mode is a default state during power up of the chip in which IOSelect is set so that all of the I/O pins in shared I/O portion 404 are coupled to received output signals from programmable logic portion 402 and to send input signals to programmable logic portion 402. Boot from Flash mode is an example of a startup mode for chip 400. In Boot from Flash mode, after Reset mode embedded processor portion 401 accesses an external Flash interface for it's boot code (e.g., using shared I/O pins). The Flash interface is required to be accessed immediately after Reset mode in Boot from Flash mode. In other modes, the Flash interface may not be required immediately after Reset mode. The default state of the IOSelect signal controls whether the Flash interface is enabled for use by programmable logic portion 402 or embedded logic portion 401. Dedicated inputs are provided that determine the initial modes of operation (e.g., Boot from Flash, Normal Configuration, etc.). Signals on these pins that are sampled during Reset mode determine the state of the register that drives the IOSelect signal for the Flash interface immediately after de-assertion of Reset mode.

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L13: Entry 1 of 5 File: USPT Jun 8, 2004

DOCUMENT-IDENTIFIER: US 6748527 B1
TITLE: Data processing system for performing software initialization

Brief Summary Text (23):
In this feature, the initialization program stored in an external memory connected through an external bus, can be transferred to a local memory while inhibiting any access request to the local memory. By this manner, in the start processing sequence when the system is powered on, the initialization program for the starting operation can be read out from the high-speed local memory, without necessity of accessing the external memory. Besides, there is no case that processors connected to the external bus start to access the external memory at once. Therefore, arbitration on the common bus becomes unnecessary, and accordingly a rapid starting operation can be performed.

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L13: Entry 2 of 5 File: USPT Jun 18, 2002

DOCUMENT-IDENTIFIER: US 6407960 B1
TITLE: Arrangement for programming selected device registers during initialization from an external memory

Brief Summary Text (10):
There is also a need for arrangement that enables users to program registers of an integrated device during device initialization without the necessity of programming an external memory, such as an EEPROM, according to a prescribed device register map that specifies programming of the registers in a prescribed sequence.

Previous Doc Next Doc Go to Doc#

L13: Entry 3 of 5 File: USPT Jul 5, 1994

DOCUMENT-IDENTIFIER: US 5327393 A
TITLE: External memory interface circuit capable of carrying out initialization of external memory with a reduced initialization time duration

Detailed Description Text (18):
When the address register 19-4 delivers the address signal having the end address of the external memory 20, the end address detecting circuit 19-6 detects the end address and delivers the end address detection signal EDS to the flip-flop circuit 19-7 as shown in a seventh line of FIG. 5. The end address detection signal EDS resets the flip-flop circuit 19-7. As a result, the flip-flop circuit 19-7 stops production of the enable signal ES. Thus, the external memory interface circuit 19 can carry out the initialization of the external memory 20 without the initialization program. In other words, the initialization is carried out without an extra time duration.

First Hit Previous Doc Next Doc Go to Doc#

Generate Collection Print

L13: Entry 4 of 5 File: JPAB Feb 4, 1994

PUB-NO: JP406028059A
DOCUMENT-IDENTIFIER: JP 06028059 A
TITLE: INITIALIZING DEVICE FOR HARDWARE SIMULATOR

PUBN-DATE: February 4, 1994

INVENTOR-INFORMATION:
NAME COUNTRY
KAWAGUCHI, HITOSHI

ASSIGNEE-INFORMATION:
NAME COUNTRY
TOSHIBA CORP

APPL-NO: JP03252495
APPL-DATE: September 30, 1991

INT-CL (IPC): G06F 1/24

ABSTRACT:

PURPOSE: To shorten the time required for memory initializing by successively starting the memory initializing of a hardware simulator from an initializing start address at the time of inputting an initializing command.

CONSTITUTION: An initializing data writing control circuit 12 reads out initializing data from an initializing data storing part 11 is which initializing data for one word are previously stored based upon a control signal (c) outputted from a sequencer 18 and then writes the read data in an area 10A to be initialized in a memory 10 together with a write signal. During the period, a down counter 20 subtracts the value of information indicating the number of words which is stored in a number of words register 16 in each fixed time interval, and when the value becomes zero, informs the sequencer 18 of the zero state by a signal (d). Thereby the sequencer 18 commands an address counter 19 to stop its operation and the output of the control signal (c) is stopped. Consequently the initializing of a specific area in the memory can be executed without transferring external initializing data and time to be required for initializing can sharply be shortened.

Previous Doc Next Doc Go to Doc#

First Hit Previous Doc Next Doc Go to Doc#

End of Result Set

Generate Collection Print

L13: Entry 5 of 5 File: JPAB Jun 11, 1990

PUB-NO: JP402151920A
DOCUMENT-IDENTIFIER: JP 02151920 A
TITLE: ELECTRONIC COMPUTER

PUBN-DATE: June 11, 1990

INVENTOR-INFORMATION:

NAME	COUNTRY
OOTA, CHINAMI	

ASSIGNEE-INFORMATION:

NAME	COUNTRY
SEIKO EPSON CORP	

APPL-NO: JP63305364
APPL-DATE: December 2, 1988

INT-CL (IPC): G06F 3/06; G06F 3/06

ABSTRACT:

PURPOSE: To initialize an external storage device so as to start operation from it again without using another external storage device by previously storing information necessary for system rise in the external storage device, and after initializing the external storage device, returning the information necessary for the system rise to the external storage device again.

CONSTITUTION: When an instruction is inputted from a keyboard device 30 so as to start the initializing operation of an external storage device 38 e.g., system information is read in by a system information reading means 36 and stored in a system information storing means 33. When the execution of initialization is confirmed, the device 38 is initialized. When the writing of the system information is confirmed, the system information stored in the means 33 is written in the device 38 by a system information writing means 35 as the system information 37. Thus, the external storage device can be initialized so as be started again without using another storage means.

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L7: Entry 1 of 5 File: USPT Sep 25, 2001

DOCUMENT-IDENTIFIER: US 6295603 B1
TITLE: Program controlled unit including a segment pointer selecting a bootstrap loader to be booted from a plurality of stored bootstrap loaders

Brief Summary Text (19):
selectively setting a content of the segment pointer prior to executing a bootstrap loading operation with the program-controlled unit.

CLAIMS:

16. A method of operating a program-controlled unit, which comprises:

storing a plurality of bootstrap loaders in a plurality of memory segments of an available memory;

booting from at least one of the memory segments from the available memory by setting a segment pointer; and

selectively setting a content of the segment pointer prior to executing a bootstrap loading operation with the program-controlled unit.

Previous Doc Next Doc Go to Doc#

L7: Entry 2 of 5 File: USPT Sep 1, 1998

DOCUMENT-IDENTIFIER: US 5802592 A
TITLE: System and method for protecting integrity of alterable ROM using digital signatures

CLAIMS:

27. A method for operating a data processing system, comprising steps of:

partitioning a bootstrap program between an unalterable read only memory device and an alterable memory device;

storing, in the alterable memory device, private key encrypted validity data representing a portion of the bootstrap program stored in the alterable memory device;

storing, in the unalterable read only memory device, a public key for decrypting the private key encrypted validity data;

in response to a triggering event, executing a portion of the bootstrap program stored in the unalterable read only memory device, the executed portion of the bootstrap program first computing validity data for at least some of the content of the alterable memory device, then using the stored public key to decrypt the private key encrypted validity data, and then comparing the decrypted validity data to the computed validity data; and

transferring control of the bootstrap program from the portion stored in the unalterable read only memory device to the portion stored in the alterable memory device only if the result of the comparison indicates that no unauthorized modifications have been made to the content of the alterable memory device.

L7: Entry 3 of 5 File: USPT Mar 5, 1996

DOCUMENT-IDENTIFIER: US 5497492 A
TITLE: System and method for loading an operating system through use of a fire system

CLAIMS:

2. A method as claimed in claim 1 wherein the computer system memory includes read-only memory (ROM) and main memory, the bootstrap program commands having a first portion and a second portion, the ROM storing the first portion of the bootstrap program commands, the method further including the step of first executing the first portion of the bootstrap program commands stored in the ROM to transfer the second portion of the bootstrap program commands and the first file system from secondary storage to main memory.

3. A method as claimed in claim 1 wherein the computer system memory includes read-only memory (ROM) and main memory, the bootstrap program commands having a first portion and a second portion, the ROM storing the first portion of the bootstrap program commands, the method further including the steps of first executing the first portion of the bootstrap program commands stored in the ROM to transfer the second portion of the bootstrap program commands from secondary storage to main memory, and using the transferred second portion of the bootstrap program commands to transfer the first file system from secondary storage to main memory.

4. A method as claimed in claim 1 wherein the computer system memory includes read-only memory (ROM) and main memory, the bootstrap program commands having a first portion and a second portion, the ROM storing the first portion of the bootstrap program commands, the method further including the steps of first executing the first portion of the bootstrap program commands stored in the ROM to transfer the first file system from secondary storage to main memory and under control of the first file system,

obtaining from the association the location in secondary storage of the second portion of the bootstrap program commands; and

transferring the second portion of the bootstrap program from the obtained location in secondary storage.

17. A computer system as claimed in claim 16 wherein the memory includes read-only memory (ROM) and main memory, the bootstrap program has a first portion and a second portion, and the first portion of the bootstrap program is stored in the ROM, the system further including: means for executing the first portion of the bootstrap program stored in the ROM to transfer the second portion of the bootstrap program and the first file system from secondary storage to main memory prior to the execution of the transferred second portion of the bootstrap program.

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

Generate Collection Print

L21: Entry 1 of 5 File: USPT Apr 27, 2004

US-PAT-NO: 6728876
DOCUMENT-IDENTIFIER: US 6728876 B1

TITLE: Method and apparatus to use non-volatile read/write memory for bootstrap code and processes by relocating write instructions

DATE-ISSUED: April 27, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kumar; Jainendra	Fremont	CA		

US-CL-CURRENT: 713/2

ABSTRACT:

A method and apparatus for significantly reducing the number and types of non-volatile memory used on a typical motherboard is disclosed. While there are typically three or more types of non-volatile memory used to support the CPU during system boot and initialization, the present invention uses only one. This allows for a significant savings in materials cost and design effort.

26 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

Previous Doc Next Doc Go to Doc#